



UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
10/813,599	03/31/2004	Gansha Wu	42339-198432	4361
26694	7590	08/04/2006	EXAMINER	
VENABLE LLP P.O. BOX 34385 WASHINGTON, DC 20045-9998			FIEGLE, RYAN PAUL	
			ART UNIT	PAPER NUMBER
			2183	

DATE MAILED: 08/04/2006

Please find below and/or attached an Office communication concerning this application or proceeding.

Office Action Summary	Application No.	Applicant(s)	
	10/813,599	WU ET AL.	
	Examiner	Art Unit	
	Ryan P. Fiegler	2183	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE _____ MONTH(S) OR THIRTY (30) DAYS, WHICHEVER IS LONGER, FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

- 1) ☒ Responsive to communication(s) filed on 11 July 2006.
- 2a) ☐ This action is FINAL. 2b) ☒ This action is non-final.
- 3) ☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

- 4) ☒ Claim(s) 1-20 is/are pending in the application.
- 4a) Of the above claim(s) _____ is/are withdrawn from consideration.
- 5) ☐ Claim(s) _____ is/are allowed.
- 6) ☒ Claim(s) 1-20 is/are rejected.
- 7) ☐ Claim(s) _____ is/are objected to.
- 8) ☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

- 9) ☐ The specification is objected to by the Examiner.
- 10) ☐ The drawing(s) filed on _____ is/are: a) ☐ accepted or b) ☐ objected to by the Examiner.
Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).
Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).
- 11) ☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

- 12) ☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).
- a) ☐ All b) ☐ Some * c) ☐ None of:
1. ☐ Certified copies of the priority documents have been received.
2. ☐ Certified copies of the priority documents have been received in Application No. _____.
3. ☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).
- * See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

- | | |
|---|---|
| 1) <input checked="" type="checkbox"/> Notice of References Cited (PTO-892) | 4) <input checked="" type="checkbox"/> Interview Summary (PTO-413) |
| 2) <input type="checkbox"/> Notice of Draftsperson's Patent Drawing Review (PTO-948) | Paper No(s)/Mail Date. _____ |
| 3) <input type="checkbox"/> Information Disclosure Statement(s) (PTO-1449 or PTO/SB/08) | 5) <input type="checkbox"/> Notice of Informal Patent Application (PTO-152) |
| Paper No(s)/Mail Date _____ | 6) <input type="checkbox"/> Other: _____ |

DETAILED ACTION

Claim Rejections - 35 USC § 103

1. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

(a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negated by the manner in which the invention was made.

2. Claims 1-20 are rejected under 35 U.S.C. 103(a) as being unpatentable over Lindwer (US Patent 6,298,434) in view of Raz et al. (Raz) (US Patent 6,606,743).

3. As per claim 1:

A method to execute an instruction on an operand stack, the method comprising:
performing a stack-state-aware translation of the instruction to code to determine an operand stack state for the instruction (Lindwer: column 11, lines 3-22) (the preprocessor moves items from registers to memory and adjusts the SP for the instructions);

dispatching the instruction according to the operand stack state for the instruction (inherent); and

executing the instruction (inherent).

Lindwer does not teach his translation including determining an entry point into shared execution code based on the stack state.

Raz teaches a language accelerator that uses memory-mapped registers as a stack (Raz: column 4, lines 7-12) among other aspects (Raz: column 1, line 60 to column 2, line 10).

Raz's accelerator translates foreign code to native code and uses memory instructions implement the stack operations on the memory-mapped stack (Raz: column 6, lines 33-38).

Thus, the stack state will determine what instructions are used to implement stack operations (e.g. an increment instruction will only pop 1 operand, while an add instruction would pop two, etc.). This, in turn, will affect the overall code length, thus affecting the entry point of the shared code (Raz: column 6, lines 24-51, emphasis on lines 45-51). The examiner is taking the position that the code is shared between the accelerator 34 and the native CPU 16, since both have access to the code.

The code is threaded (Raz: column 4, lines 29-31). The implementation and advantages of multithreading is well known in the art and would have been obvious to one of ordinary skill in the pertinent art at the time of the applicant's invention.

Raz states that his method increases the speed at which Java code is executed (Raz: column 2, lines 11-12). In addition, Raz's method can be readily implemented in any processor (Raz: column 13-19), while Lindwer's cannot.

Therefore, it would have been obvious to one of ordinary skill in the pertinent art at the time of the applicant's invention to implement Lindwer's stack system as in Raz's to be able to execute Java code faster and to be able to implement the method in any processor.

4. As per claim 2:

The method according to claim 1, said performing comprising:

determining a number of operands on the operand stack before the instruction is executed (Lindwer: column 11, lines 3-22) (It is inherent that this step will be taken in moving items from registers to memory and adjusting the SP for the instructions);

determining a number of operands on the operand stack after the instruction is executed based on a number of operands that the instruction consumes and a number of operands that the instruction produces (Lindwer: column 11, lines 3-22); and

inferring a number of shift operations required after execution of the instruction to maintain top-of-stack elements (Lindwer: column 11, lines 3-22).

5. As per claim 3:

The method according to claim 2, wherein the number of shift operations required after execution of the instruction is based on the number of operands on the operand stack before the instruction is executed and the number of operands on the operand stack after the instruction is executed (Lindwer: column 11, lines 15-22).

6. As per claim 4:

The method according to claim 2, wherein the number of shift operations required after execution of the instruction is inferred based on a static lookup table (Lindwer: column 6, lines 39-47) (The translation is based on a static table. Through the table, it is known how many operands will be used and how many will be placed back on the stack, and based on that is how many items are transferred to memory.).

7. As per claim 5:

The method according to claim 1, wherein the operand stack is a mixed-register stack (Lindwer: column 11, lines 15-22).

8. As per claim 6:

The method according to claim 1, wherein the operand stack state comprises a number of shift operations to maintain top-of-stack elements of the operand stack after the execution of the instruction (Lindwer: column 11, lines 15-22).

9. As per claim 7:

The method according to claim 6, wherein the top-of-stack elements comprise a register stack (Lindwer: column 11, lines 15-22).

10. As per claim 8:

The method according to claim 1, further comprising:
refilling the operand stack (Lindwer: column 11, lines 15-22) (The items are moved based on what will be overwritten, meaning that values pushed on the stack from the routine will refill the register part of the stack.).

11. As per claim 9:

A system comprising:
an operand stack to execute an instruction (Lindwer: column 11, lines 3-5); and
an interpreter to determine a state of the operand stack, translate the instruction into threaded code, and dispatch the instruction based on the state of the operand stack (Lindwer: column 11, lines 3-22) (the preprocessor is the interpreter).

Lindwer does not teach his interpreter determining an entry point into shared execution code based on the stack state.

Raz teaches a language accelerator that uses memory-mapped registers as a stack (Raz: column 4, lines 7-12) among other aspects (Raz: column 1, line 60 to column 2, line 10).

Raz's accelerator translates Java code, in some embodiments, to native code and uses memory instructions implement the stack operations on the memory-mapped stack (Raz: column 6, lines 33-38).

Thus, the stack state will determine what instructions are used to implement stack operations (e.g. an increment instruction will only pop 1 operand, while an add instruction would pop two, etc.). This, in turn, will affect the overall code length, thus affecting the entry point of the shared code (Raz: column 6, lines 24-51, emphasis on lines 45-51). The examiner is taking the position that the code is shared between the accelerator 34 and the native CPU 16, since both have access to the code.

The code is threaded (Raz: column 4, lines 29-31). The implementation and advantages of multithreading is well known in the art and would have been obvious to one of ordinary skill in the pertinent art at the time of the applicant's invention.

Raz states that his method increases the speed at which Java code is executed (Raz: column 2, lines 11-12).

In addition, Raz's method can be readily implemented in any processor (Raz: column 13-19), while Lindwer's cannot.

Therefore, it would have been obvious to one of ordinary skill in the pertinent art at the time of the applicant's invention to implement Lindwer's stack system as in Raz's

Art Unit: 2183

to be able to execute Java code faster and to be able to implement the method in any processor.

12. As per claim 10:

The system according to claim 9, wherein the operand stack is a mixed stack comprising a register stack and a memory stack (Lindwer: column 11, lines 15-22).

13. As per claim 11:

The system according to claim 10, wherein the register stack comprises at least one register to hold at least one respective top element of the stack and the memory stack comprises a contiguous memory region to hold the remaining elements of the operand stack (Lindwer: column 3, lines 15-22).

14. As per claim 12:

A machine accessible medium containing program instructions that, when executed by a processor, cause the processor to perform a series of operations comprising:

translating a virtual machine instruction into threaded code based on an operand stack state of the virtual machine instruction (Lindwer: column 11, lines 3-22);

dispatching the virtual machine instruction according to the operand stack state (inherent); and

executing the instruction (inherent).

Lindwer does not teach his translation including determining an entry point into shared execution code based on the stack state.

Raz teaches a language accelerator that uses memory-mapped registers as a stack (Raz: column 4, lines 7-12) among other aspects (Raz: column 1, line 60 to column 2, line 10).

Raz's accelerator translates Java code, in some embodiments, to native code and uses memory instructions implement the stack operations on the memory-mapped stack (Raz: column 6, lines 33-38).

Thus, the stack state will determine what instructions are used to implement stack operations (e.g. an increment instruction will only pop 1 operand, while an add instruction would pop two, etc.). This, in turn, will affect the overall code length, thus affecting the entry point of the shared code (Raz: column 6, lines 24-51, emphasis on lines 45-51). The examiner is taking the position that the code is shared between the accelerator 34 and the native CPU 16, since both have access to the code.

The code is threaded (Raz: column 4, lines 29-31). The implementation and advantages of multithreading is well known in the art and would have been obvious to one of ordinary skill in the pertinent art at the time of the applicant's invention.

Raz states that his method increases the speed at which Java code is executed (Raz: column 2, lines 11-12).

In addition, Raz's method can be readily implemented in any processor (Raz: column 13-19), while Lindwer's cannot.

Therefore, it would have been obvious to one of ordinary skill in the pertinent art at the time of the applicant's invention to implement Lindwer's stack system as in Raz's

Art Unit: 2183

to be able to execute Java code faster and to be able to implement the method in any processor.

15. As per claim 13:

The machine accessible medium according to claim 12, wherein the threaded code is based on an entry point into shared execution code (Lindwer: column 11, lines 3-22) (it is an entry into a subroutine) (Raz: column 6, lines 45-51).

16. As per claim 14:

The machine accessible medium according to claim 12, further containing program instructions that, when executed by the processor cause the processor to perform further operations comprising:

determining a number of operands that are present on an operand stack at a time before the virtual machine instruction is executed (Lindwer: column 11, lines 3-22) (It is inherent that this step will be taken in moving items from registers to memory and adjusting the SP for the instructions);

determining a number of operands that are present on the operand stack at a time after the virtual machine instruction is executed (Lindwer: column 3, lines 3-22);

and

inferring a number of shift operations required to maintain top-of-stack elements after the virtual machine instruction is executed (Lindwer: column 3, lines 3-22).

17. As per claim 15:

The machine accessible medium according to claim 13, wherein the wherein the number of shift operations required after execution of the instruction is based on the

Art Unit: 2183

number of operands present on the operand stack at a time before the instruction is executed and the number of operands present on the operand stack at a time after the instruction is executed (Lindwer: column 11, lines 15-22).

18. As per claim 16:

The machine accessible medium according to claim 13, wherein the number of shift operations required after execution of the instruction is inferred based on a static lookup table (Lindwer: column 6, lines 39-47) (The translation is based on a static table. Through the table, it is known how many operands will be used and how many will be placed back on the stack, and based on that is how many items are transferred to memory.).

19. As per claim 17:

The machine accessible medium according to claim 12, wherein the operand stack state comprises a number of shift operations to maintain top-of-stack elements of an operand stack after execution of the virtual machine instruction (Lindwer: column 11, lines 15-22).

20. As per claim 18:

The machine accessible medium according to claim 17, wherein the top-of-stack elements comprise a register stack (Lindwer: column 11, lines 15-22).

21. As per claim 19:

The machine accessible medium according to claim 12, further containing program instructions that, when executed by the processor cause the processor to perform further operations comprising:

execute a number of shift operations to replace top-of-stack elements to an operand stack (Lindwer: column 11, lines 15-22) (The items are moved based on what will be overwritten, meaning that values pushed on the stack from the routine will refill the register part of the stack.).

22. As per claim 20:

The machine accessible medium according to claim 19, wherein the number of shift operations is based on a number of elements on the operand stack that are consumed by the virtual machine instruction and a number of elements that are produced by the virtual machine instruction (Lindwer: column 11, lines 15-22) (The items are moved based on what will be overwritten, meaning that values pushed on the stack from the routine will refill the register part of the stack.).

Response to Arguments

23. In a personal interview, the examiner agreed that Lindwer alone could not teach determining an entry point into shared execution code *based* on the stack state.

Consequently, the examiner has withdrawn the finality of the previous office action; this office action is non-final.

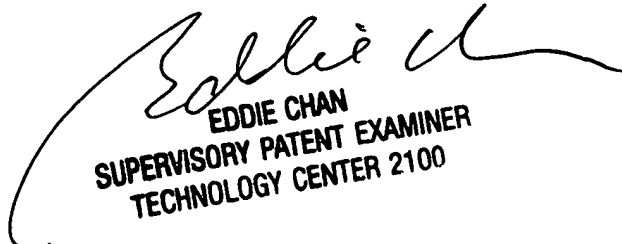
Conclusion

Any inquiry concerning this communication or earlier communications from the examiner should be directed to Ryan P. Fiegle whose telephone number is 571-272-5534. The examiner can normally be reached on M-F 8-4:30.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Eddie Chan can be reached on 571-272-4162. The fax phone number for the organization where this application or proceeding is assigned is 571-273-8300.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see <http://pair-direct.uspto.gov>. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free). If you would like assistance from a USPTO Customer Service Representative or access to the automated information system, call 800-786-9199 (IN USA OR CANADA) or 571-272-1000.

Ryan P Fiegle
Examiner
Art Unit 2183



EDDIE CHAN
SUPERVISORY PATENT EXAMINER
TECHNOLOGY CENTER 2100